

Mathematical Modeling of General Inaccurate Adders

Zvi M. Kedem*

Kirthi Krishna Muntimadugu†

2016-06-07

Abstract

Inaccurate circuits make possible the conservation of limited resources, such as energy. But effective design of such circuits requires an understanding of resulting tradeoffs between accuracy and design parameters, such as voltages and speed of execution. Although studies of tradeoffs have been done on specific circuits, the applicability of those studies is narrow. This paper presents a comprehensive and mathematically rigorous method for analyzing a large class of inaccurate circuits for addition. Furthermore, it presents new, fast algorithms for the computation of key statistical measures of inaccuracy in such adders, thus helping hardware architects explore the design space with greater confidence.

1 Introduction

Each day, situations arise in which accuracy is traded off for some other objective. For example, noticeable changes occur in the quality video that is streamed online as the amount of bandwidth that can be allocated changes. This is an instance of software-controlled, inaccurate computation. A similar notion can be applied to hardware as well, where the energy that can be allocated to computations is constrained. In this paper, we study inaccurate circuits, specifically circuits for binary additions that may produce incorrect sums for some inputs. Such incorrect sums may be the result of the outputs being read before the computation had a chance to complete (such as approximate adders which violate the critical path, studied e.g. by Kedem et al. [1]), or they may be the result of the hardware not being designed to be “fully” correct (such as pruned adders studied e.g. by Lingamneni et al. [2]). In brief, such circuits

*Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, New York, NY 10012-1180, USA.

†Austin, TX, USA.

are designed and operated in a way that trades off accuracy for the usage of a limited resource such as energy. Particularly because of energy limitations, such circuits are the subject of rapidly growing interest.

An inaccurate ripple-carry adder (RCA) and the expected error under various assignments of voltages to its gates were studied in [1], and we refer interested readers to that paper and its references for motivation, additional background, and related work, beyond what is described here and what is required for understanding the results of this paper. To emphasize that we are considering inaccurate circuits for addition, we will refer to such circuits as *pseudo-adders*. Of course, the set of correct adders is a subset of the set of pseudo-adders.

The focus of the paper is exclusively on the mathematical properties of the outputs produced by pseudo-adders. The mathematical properties we study are fundamental to the analysis of important factors in the design of pseudo-adders, such as delays, layouts, voltage assignments, and the energy consumed during the operation.

In order to meaningfully explore the design space, it is necessary to have metrics for the evaluation of alternative designs. One such metric would be, of course, the inherent inaccuracy of a pseudo-adder. A useful measure for this is the expected error, and [1] developed a fast way for computing it for one type of overclocked RCA. That paper, however, had limited applicability to other adders because the technique that it developed was targeted to RCAs and crucially relied on specific characteristics of RCAs.

Results presented here go much beyond that paper. First, we present a general method capable of handling a large class of pseudo-adders. As an example, we show the applicability of our method to the highly parallel Kogge-Stone adder. The set of statistical measure is also richer. We provide not only an algorithm for computing the expected error but also algorithms for computing the mean squared error and the maximum error. All of the algorithms are very fast.

To compute the statistical measures listed above, ostensibly one needs to consider the results of computing $a + b$ for all possible pairs of inputs (a, b) . As there are 2^{2n} of them for an n -bit adder, such an approach would be impossible for other than rather modest values of n . One could optimize somewhat by for example only considering pairs for which $a \leq b$, but savings of this type are not significant.

As was done in [1], we reduce the computational effort required by only analyzing the errors in carry chains, thus, in effect having to deal with only $n(n + 1)/2$ pairs of inputs while still computing the statistical measures valid for the complete set of input pairs. In consequence, the number of input pairs that need to be considered is reduced dramatically. For $n = 64$, instead of considering 340.282×10^{36} pairs, we need to consider only 2.08×10^3 pairs, a reduction by 35 orders of magnitude.

For a general background on computer arithmetic, see e.g., Parhami [3]. To facilitate comparison with [1] by interested readers, we will use mainly its notation, but we will modify it where it makes the presentation of our results clearer.

The rest of the presentation is organized as follows. We first briefly review the

notion of carry chains and formulate our model and the class of pseudo-adders we will analyze. We then review the results obtained in [1], also showing when they are *not* applicable. We then proceed to our more general treatment, formulating and proving the characteristics of the errors that can occur in them. This leads to the formulation of fast algorithms for the computation of the statistical measures of interests. We show an example of an interaction of errors for a pair of inputs in a Kogge-Stone-based pseudo-adder. We conclude by recapitulating our results.

2 Inputs and carry chains

The fundamental issue in the design of fast adders is the possible presence of carry chains. Carry chains were studied as early as the 19th century by Babbage [4] for decimal adders and in 1946 by Burks et al. [5] for binary adders.

We will consider the addition of two numbers a and b from the range $[0, \dots, 2^n - 1]$, for some fixed n . Using standard binary representation, such numbers are in one-to-one correspondence with n -bit vectors. It will, however, be more convenient to consider input vectors of length $n + 1$: $\mathbf{a} = \langle a_n a_{n-1} \dots a_0 \rangle$ and $\mathbf{b} = \langle b_n b_{n-1} \dots b_0 \rangle$, with $a_n = b_n = 0$. Given two numbers a and b , two sequences \mathbf{c} and \mathbf{s} of length $n + 1$ are defined by

$$\begin{aligned} c_k &= a_{k-1}b_{k-1} \vee a_{k-1}c_{k-1} \vee b_{k-1}c_{k-1} \\ s_k &= a_k \oplus b_k \oplus c_k \end{aligned} \tag{1}$$

for $k = 0, 1, \dots, n$, with $c_0 = 0$. Thus, \mathbf{s} corresponds to the sum $a + b$ and c_k is the carry needed to compute s_k , for $k = 1, \dots, n$. Of course, c_k is a function of $\langle a_{k-1} a_{k-2} \dots a_0 \rangle$ and $\langle b_{k-1} b_{k-2} \dots b_0 \rangle$. In the rest of the paper, when a and b are understood from the context, we may omit them.

We *adapt* a standard definition of a carry chain:

Definition 1. Given n -bit numbers a and b , there is a *carry chain starting at i and ending at j* for $1 \leq i \leq j \leq n$ when

1. $a_{i-1} = b_{i-1} = 1$
2. $a_k \neq b_k$ for $i \leq k \leq j - 1$
3. $a_j = b_j$.

We will refer to such a carry chain *i - j carry chain* and also say that it is *from i to j* . We will say that the pair (a, b) *generates* this carry chain. The set of all the carry chains generated by (a, b) will be denoted by $\mathcal{C}(a, b)$.

Note that our terminology for carry chains is slightly different from the more common one in which a carry chain is a sequence of positions that generate and propagate a carry. That is, using our notation, in the common definition the carry chain starts at position $i - 1$ and it ends in position $j - 1$. We chose our notation, as it is more convenient to focus *not* on the positions from which there are *outgoing* carries of 1, but on the positions into which there are *incoming* carries of 1 because those are the positions where errors in the sum may occur.

Fixing (a, b) and using the notation

$$C_{ij}(a, b) = \begin{cases} \top & \text{if there is a carry chain from } i \text{ to } j \\ \perp & \text{otherwise,} \end{cases}$$

two well known and easily proved [1] properties are

Lemma 1. *If $C_{ij}(a, b) = C_{pq}(a, b) = \top$ and $(i, j) \neq (p, q)$, then the intervals $[i, j]$ and $[p, q]$ are disjoint, or equivalently, the two carry chains do not overlap.*

Lemma 2. *If $C_{ij}(a, b) = \top$ and $s = a + b$, then $s_k = 0$, for $k = i, \dots, j - 1$, and $s_j = 1$.*

3 A model for pseudo-adders

We will model circuits under consideration by directed acyclic graphs (DAGs) with $2n + 1$ input (source) vertices and $n + 1$ output (sink) vertices. The two inputs \mathbf{a} and \mathbf{b} and the input carry c_0 are supplied at the input vertices at time $t = 0$ and the sum \mathbf{s} is supposed to be read from the output vertices at some later time T . However, at time $t = T$, the output vertices do not necessarily contain the correct sum \mathbf{s} but some vector \mathbf{s}' , which may differ from \mathbf{s} , and in general the correct \mathbf{s} may not ever be available at the outputs.

We can write an equation analogous to (1):

$$s'_k = a_k \oplus b_k \oplus c'_k,$$

where c'_k for $k = 1, 2, \dots, n$ is a function of $\langle a_{k-1} a_{k-2} \dots a_0 \rangle$ and $\langle b_{k-1} b_{k-2} \dots b_0 \rangle$, just as c_k was. In the completely general setting, no assumptions can be made about the relation between c_k and c'_k , so interesting results are not likely to be possible. We thus restrict our treatment to a useful subset of cases.

Definition 2. A pseudo-adder will be called *conservative* if for all k , $c'_k \leq c_k$, that is whenever $c_k = 0$ then also $c'_k = 0$. Therefore, there will be no spurious carries.

A situation for which such pseudo-adders appear naturally is one in which the circuit itself is a correct adder, but it is overlocked. Then, it may be the case that for some k the circuit's "default" value for c_k is 0, but for some inputs $c_k = 1$ and that information is not properly processed by time $t = T$. In consequence, $c'_k = 0$ and $s'_k \neq s_k$. However, there are conservative pseudo-adders that are not simply an overlocked version of correct adders but are incorrect by their architecture.

We will restrict ourselves to conservative pseudo-adders. We make two technical assumptions, which may seem obvious but they need to be explicitly stated in order to define the error contributed by a carry chain.

Assumption 1. Because we are dealing with conservative pseudo-adders, errors can only occur when for some k , $c_k = 1$ but $c'_k = 0$. Therefore, $i \leq k \leq j$, for

some i - j carry chain., As $c_k = 1$ is determined independently of the values of $\langle a_{i-2}, a_{i-3} \dots a_0 \rangle$ and $\langle b_{i-2}, b_{i-3} \dots b_0 \rangle$, we assume that $c'_k = 0$ is also determined independently of those values. So we assume that the input values in the positions *lower* than $i - 1$ cannot influence what happens during the propagation of the carry generated *at* $i - 1$.

Assumption 2. Of course the addition $a + b$ is commutative but one can envision conservative pseudo-adder circuits in which the behavior of the computation is not commutative. Imagine a correct adder design which however has a fabrication error: it ignores the supplied value of a_0 and always propagates it as 0. In such a circuit, $0 + 1$ will result in 1 but $1 + 0$ will result in 0. Since we are interested in the properties of a carry chain that are independent of the specific input pair generating it, we will assume that c' is the same for $a + b$ and $b + a$.

Consider any carry chain C and a pair of inputs (a, b) that generates it. Define a pair of inputs (a^C, b^C) as follows: $a_0^C = a_1^C = \dots = a_{i-2}^C = 0$, $a_{i-1}^C = a_{i-1}$, $a_i^C = a_i$, \dots , $a_{j-1}^C = a_{j-1}$, $a_j^C = a_{j+1}^C = \dots = a_{n-1}^C = 0$, and analogously for b . In effect we have isolate the carry chain by “zeroing out” positions irrelevant to its appearance.

We can now state

Definition 3. Let C be a carry chain and (a, b) any pair of inputs generating C . For the associated (a^C, b^C) let s and s' denote respectively the true sum and the sum computed by the pseudo-adder. Then we define the error of the carry chain as

$$\text{EC}(C) = s - s'$$

(although more conventionally one might use $s' - s$).

Note the importance of the two assumptions above. They allowed us to define the error of the carry chain by considering *any pair* of inputs generating the chain for defining the chain’s error. Our being able to do that made the concept of the error of the carry chain meaningful.

The implications for further development can be stated succinctly:

Lemma 3. *In a conservative pseudo-adder errors are determined by carry chains.*

Proof. The lemma holds because if two sums $a^{(1)} + b^{(1)}$ and $a^{(2)} + b^{(2)}$ have carry chains in the same positions, then the *computed* sums for them are the same. \square

4 Reducing from errors in sums to errors in carry chains

4.1 Errors and carry chains

The material in this section contains mostly a general reformulation of the material previously presented in [1] for a more restrictive case, but we start with

		Position							
		7	6	5	4	3	2	1	0
a	→	0	0	1	0	1	0	1	0
b	→	0	0	0	1	1	1	0	1
c	→	0	1	1	1	1	1	1	0
s	→	0	1	0	0	1	0	0	1
a	→	0	0	1	0	1	0	1	0
b	→	0	0	0	1	1	1	0	1
c'	→	0	1	0	0	0	1	1	0
s'	→	0	1	1	1	0	0	0	1

Figure 1: Carry and sum bits for an addition in an 8-bit adder. The upper block shows correct addition and the lower block an erroneous one. Carry chains are highlighted.

an example of the case that could not be treated there.

Example 1. We consider the computation of $a + b$, where $\mathbf{a} = \langle 001010110 \rangle$ and $\mathbf{b} = \langle 000111010 \rangle$, generating the 2-4 carry chain and the 5-7 carry chain. The situation is depicted in Fig. 1. The top block illustrates the correct computation and the bottom block an incorrect one, which can take place in a pseudo-adder. The various values are $\mathbf{c} = \langle 011111100 \rangle$, $\mathbf{s} = \langle 010010000 \rangle$, $\mathbf{c}' = \langle 010001100 \rangle$, and $\mathbf{s}' = \langle 011100001 \rangle$. Note that as $c'_k \leq c_k$ for every k , this is a conservative pseudo-adder (at least for these inputs). Note also that the errors contributed by the two carry chains are respectively -96 and $+16$, so the errors contributed by carry chains can be both negative and positive. In Section 6, we will see exactly how the situation in the lower block can occur in an KSA-based pseudo-adder. This completes the example.

The algorithm in [1] was limited in *not accounting* for the possibility of negative errors in carry chains; it considered only overclocked RCAs, where such errors cannot occur, as we discuss further in Section 4.2.

Returning to a general conservative pseudo-adder, we examine more closely the difference between s_k and s'_k for some fixed k . We need to consider several cases:

1. If $c_k \neq c'_k$, then $c_k = 1$, $c'_k = 0$, and k is such that for some (i, j) , $i \leq k \leq j$, we have $C_{ij}(a, b) = \top$. Then there are two cases:
 - (a) If $k < j$, then $s_k = 0$, $s'_k = 1$, and $s_k - s'_k = -1$.
 - (b) If $k = j$, then $s_k = 1$, $s'_k = 0$, and $s_k - s'_k = +1$.
2. If $c_k = c'_k$ then $s_k - s'_k = 0$.

We are interested in statistical properties of errors in the computations by pseudo-adders. A natural measure to consider is the absolute value of the average error.

The absolute error in the addition $a + b$ is

$$\text{Er}(a, b) = |s(a, b) - s'(a, b)|$$

and the expectation of the absolute value of the error is

$$\text{Er}_{\text{avg}} = \frac{1}{2^{2n}} \sum_{a,b} \text{Er}(a, b), \quad (2)$$

assuming the uniform distribution as we do here. So we need to compute *the sum of the absolute values of the errors* (SAE), that is

$$\text{SAE} = \sum_{a,b} \text{Er}(a, b), \quad (3)$$

a sum of 2^{2n} terms, which is an impractical task for other than relatively modest values of n .

We will replace (3) by a certain sum involving errors in the $n(n+1)/2$ carry chains possible in the addition of n -bit numbers and we will use C as a variable varying over this set of all the carry chains. If C contributes an error in the sum of a pair of inputs, we refer to C as an *error-contributing chain* for those inputs.

Because the carry chains generated by a pair of inputs do not overlap, we can rewrite (3) as

$$\text{SAE} = \sum_{a,b} \left| \sum_{C \in \mathcal{C}(a,b)} \text{EC}(C) \right|, \quad (4)$$

where $\mathcal{C}(a, b)$ is the set of the carry chains generated by (a, b) .

4.2 The special case of the fine-grained overlocked RCA

We now turn our attention to a pseudo-adder obtained from the standard RCA by *fine-grained overlocking*. The modules of the RCA (at whatever level of granularity is technologically feasible) are supplied with different voltages. Even identical modules, such as full adders, may have different voltages supplied in different parts of the circuit. In consequence, identical modules in different parts of the adder might have different delays. If operated at a fixed frequency, such an adder may not always operate correctly since sometimes the output might be read faster than the critical path delay. These voltages, and hence the delays, may be “distributed” to optimize some accuracy measure as shown in [1].

In specific pseudo adders studied in this Section 4.2, errors in carry chains have a property that greatly simplifies their treatment. Intuitively, because for each carry chain, the carry “ripples” through sequentially, the error contributed by a carry chain can only result in an output sum that is smaller than the true one. Specifically,

Theorem 1. *In a pseudo-adder based on the ripple-carry adder*

$$\text{EC}(C) \geq 0 \quad (5)$$

for every carry chain C .

Proof. Consider some i - j carry chain C for which $\text{EC}(C) \neq 0$. The correct values are $s_i = s_{i+1} = \dots = s_{j-1} = 0$ and $s_j = 1$. Due to the structure of the RCA, $\text{EC}(C) \neq 0$ because the carry has not yet propagated to positions $k+1, k+2, \dots, j$, for some $k+1 \leq j$. Then $s'_i = s'_{i+1} = \dots = s'_k = 0$, $s'_{k+1} = \dots = s'_{j-1} = 1$, and $s'_j = 0$. Therefore, $s - s' = 2^j - \sum_{\ell=k+1}^{j-1} 2^\ell > 0$. \square

In consequence, of course, $|\text{EC}(C)| = \text{EC}(C)$ and therefore

$$\sum_{a,b} \left| \sum_{C \in \mathcal{C}(a,b)} \text{EC}(C) \right| = \sum_{a,b} \sum_{C \in \mathcal{C}(a,b)} \text{EC}(C).$$

Let $\nu(C)$ denote the number of input pairs (a, b) for which C appears in $a + b$. Then we can change the order of summation and therefore

$$\sum_{a,b} \sum_{C \in \mathcal{C}(a,b)} \text{EC}(C) = \sum_C \text{EC}(C) \cdot \nu(C).$$

Crucially relying on (5) in studying a specific type of pseudo-adder based on the RCA, [1] used

$$\text{SAE} = \sum_C \text{EC}(C) \cdot \nu(C) \quad (6)$$

to compute the absolute value of the average error (although actually the result of interest was phrased there equivalently using probabilities of the errors and not directly using the number of input pairs that generate them). However, for circuits other than RCAs, (5) *does not necessarily hold* and in such cases (6) *will not produce the correct SAE*.

4.3 The case of general conservative pseudo-adders

To treat general pseudo-adders, a different procedure for computing (4) must be established, which we do next. Later we apply it in the example of the KSA for which, indeed (5) does not hold.

The goal is to dispense with absolute values. As the development is somewhat technical, we start with an example to provide intuitive understanding behind (8).

Example 2. The example is not realistic but is well suited for providing intuition. Assume that we pick some carry chain C and we want to know what the contribution of the error in C to the sum of absolute errors is. Its error is $\text{EC}(C) = -6$, and C is generated by only three input pairs each of which generates exactly one additional and distinct error-contributing chain. Those additional chains C_1 , C_2 , and C_3 contribute respectively errors of $+13$, -14 , and

+15. Then the total errors for those inputs are +7, -20, and +9. As the absolute values of these errors are larger than that of $\text{EC}(C)$, the signs of these errors determine the signs of the total error for their respective input pairs. Hence, we will designate these three carry chains as dominating ones for their respective input pairs.

We now note that we can obtain the absolute value of the sum of errors for some input by taking the error of the individual chains, multiplying each by the signum of the error of the dominating chain, and adding these terms together. Indeed, the three signs are +1, -1, and +1, and therefore

$$\begin{aligned} |-6 + 13| &= (+1)(-6) + (+1)(+13) = +7 \\ |-6 - 14| &= (-1)(-6) + (-1)(-14) = +20 \\ |-6 + 15| &= (+1)(-6) + (+1)(+15) = +9. \end{aligned}$$

We now compute the contribution of $\text{EC}(C)$ to SAE. It is

$$(+1)(-6) + (-1)(-6) + (+1)(-6) = (-6)(+2 - 1), \quad (7)$$

and the SAE is

$$(+1)(+13) + (-1)(-14) + (+1)(+15) + (-6)(+2 - 1).$$

Note that 2 was the number of the input pairs generating C in which the dominating chain was positive and 1 was the number of the input pairs generating C in which the dominating chain was negative. The numbers +2 and -1 on the right-hand side of (7) correspond to $\nu^+(C)$ and $\nu^-(C)$ in (8). This completes the example.

We now proceed to the general case. Assume that $\mathcal{C}(a, b)$ has at least one error-contributing carry chain for input pair (a, b) . We take the *leftmost* (highest position numbers) chain contributing an error, refer to it as the *dominating carry chain* for (a, b) , and denote it by $C_L(a, b)$. The following lemma is pivotal to the subsequent development as it states that the sign of the total error for a pair of inputs is determined by the leftmost chain that pair generates.

Lemma 4.

$$\text{sgn}\left(\sum_{C \in \mathcal{C}(a, b)} \text{EC}(C)\right) = \text{sgn}\left(\text{EC}(C_L(a, b))\right),$$

where, as usual, sgn denotes the signum function.

Proof. The idea behind the proof is that if k is the largest position in which there is an error, it is also the largest position in which there is an error in the positions of the dominating carry chain.

Let \mathbf{u} and \mathbf{v} be $(n+1)$ -bit vectors such that for some k , $0 \leq k \leq n$, $u_n = v_n$, $u_{n-1} = v_{n-1}, \dots, u_{k+1} = v_{k+1}$ but $u_k = 1$ and $v_k = 0$. Then, obviously, $u > v$.

Consider now inputs a and b and the corresponding s and s' such that $s \neq s'$. Let k be the leftmost position in which $s_k \neq s'_k$. Then $C_L(a, b)$ is

an i - j carry chain such that $i \leq k \leq j$. We will write C_L for $C_L(a, b)$ in this proof. Recalling a notation from Definition 3, let $s(C_L)$ and $s'(C_L)$ denote respectively the correct and the computed sum for the input pair (a^{C_L}, b^{C_L}) . Then, $s(C_L) - s'(C_L) = \text{EC}(C_L)$.

Position k is also the leftmost position in which $s(C_L)$ and $s'(C_L)$ are different, and furthermore $s_k(C_L) = s_k$ and $s'_k(C_L) = s'_k$. Therefore $s > s'$ if and only if $s(C_L) > s'(C_L)$, and as $s - s' = \sum_{C \in \mathcal{C}(a, b)} \text{EC}(C)$, the lemma is proved. \square

As we will see next, the lemma allows us to replace the absolute sum in (4) by an appropriate collection of signed (multiplied by +1 or -1) $\text{EC}(C)$'s. Let

$$\widehat{\text{sgn}}(a, b) = \begin{cases} 0 & \text{if } \mathcal{C}(a, b) = \emptyset \\ \text{sgn}(\text{EC}(C_L(a, b))) & \text{otherwise.} \end{cases}$$

Recall that the absolute value of the error contributed by C to the computation of $a + b$ is the same for all pairs of inputs (a, b) which generate C . Let

$$\nu^\sigma(C) = \left| \{ (a, b) \mid C \in \mathcal{C}(a, b) \wedge \widehat{\text{sgn}}(a, b) = \sigma \} \right|,$$

where the outer bars denote the cardinality of the set. (Note that ν^σ is *not* ν to the power of σ , but rather either ν^+ or ν^- .) The number $\nu^\sigma(C)$ is the number of input pairs that generate C and in which the sign of the dominating chain is σ .

We are now ready to handle the absolute values in (4) using $\widehat{\text{sgn}}(a, b)$. The key property follows. Consider some pair of inputs (a, b) for which there is an error in the pseudo-adder. By Lemma 4, the sign of the total error in $a + b$ is the sign of the corresponding dominating chain, $C_L(a, b)$. Therefore, to obtain the absolute value of the total error, we can multiply the total error by the sign of the dominating chain, that is by $\widehat{\text{sgn}}(a, b)$. This leads to the following.

Theorem 2.

$$\text{Er}_{\text{avg}} = \frac{1}{2^{2n}} \sum_C \text{EC}(C) \cdot (\nu^+(C) - \nu^-(C)). \quad (8)$$

Proof. We first note that

$$\begin{aligned} \left| \sum_{C \in \mathcal{C}(a, b)} \text{EC}(C) \right| &= \widehat{\text{sgn}}(a, b) \cdot \sum_{C \in \mathcal{C}(a, b)} \text{EC}(C) \\ &= \sum_{C \in \mathcal{C}(a, b)} \widehat{\text{sgn}}(a, b) \cdot \text{EC}(C). \end{aligned}$$

Positions	Inputs
$0, \dots, i-2$	00, 01, 10, 11
$i-1$	11
$i, \dots, j-1$	01, 10
j	00, 11
$j+1, \dots, p-2$	00, 01, 10, 11
$p-1$	11
$p, \dots, q-1$	01, 10
q	00
$q+1, \dots, n-1$	00, 01, 10

Figure 2: Input conditions for i - j carry chain and for the corresponding leftmost p - q carry chain. The left column lists the positions and the right column lists the pairs of inputs allowed in the positions. Note that possibly $q = n$ and then the last row does not apply. The relation between a carry chain and the probability of an input pair producing it was already studied in [5].

Then, having dispensed with the absolute values, we have

$$\begin{aligned}
\sum_{a,b} \left| \sum_{C \in \mathcal{C}(a,b)} \text{EC}(C) \right| &= \sum_{a,b} \sum_{C \in \mathcal{C}(a,b)} \widehat{\text{sgn}}(a,b) \cdot \text{EC}(C) \\
&= \sum_C \sum_{\{(a,b) \mid C \in \mathcal{C}(a,b)\}} \widehat{\text{sgn}}(a,b) \cdot \text{EC}(C) \\
&= \sum_C \text{EC}(C) \cdot (\nu^+(C) - \nu^-(C)),
\end{aligned}$$

and the theorem immediately follows. \square

In the simple case of the RCA, $\nu^-(C) = 0$ and we can write ν for ν^+ , simplifying the equation and getting (6).

To compute (2) fast, it is enough to compute all the terms in (8) fast. For every C , one needs to compute $\text{EC}(C)$, and we return to this later, assuming here that we have these values for the pseudo-adder under consideration.

4.4 Computing fast the total effect of a carry chain error

We need to compute $\nu^+(C)$ and $\nu^-(C)$ where C is an error-contributing i - j carry chain and $1 \leq i \leq j \leq n$. If C is the leftmost error-contributing chain generated by some input pair (a,b) , then $\widehat{\text{sgn}}(a,b) = \text{sgn}(\text{EC}(C))$. This is the simpler case, so we will consider here the case where C is not the leftmost error-contributing chain and let p - q carry chain be the leftmost error-contributing chain generated by (a,b) . For this situation, $n \geq q \geq p > j \geq i \geq 1$ and the conditions in Fig. 2 hold. The number of input configurations producing such a pair of carry chains is

1. If $n = q$ then $2^{n-p}2^{j-i}4^{(p-1)-(j-i+1)}$.
2. If $n > q$ then $3^{n-q+1}2^{q-p}2^{j-i}4^{(p-1)-(j-i+1)}$.

Keeping i and j fixed, we consider all the possible values for p and q . For some inputs, the i - j carry chain is the leftmost error-contributing chain, and for some inputs there is another error-contributing chain, p - q carry chain, and the latter is the leftmost such carry chain. By examining all the cases, we can compute $\nu^+(C)$ and $\nu^-(C)$. Varying the i - j carry chain over $1 \leq i \leq j \leq n$ we treat all the carry chains and from this we compute the needed Er_{avg} , using (2).

It is immediately clear how to compute (8) in $\Theta(n^4)$ steps by considering all relevant quadruples (i, j, p, q) . We sketch next how to do that in $\Theta(n^2)$ steps.

Fix now an error-contributing i - j carry chain C , where $1 \leq i \leq j \leq n$. Of course we know the number of input pairs that generate it. To compute the needed ν^+ and ν^- we need to consider leftmost error-contributing carry chains contained in the interval $[j+1, j+2, \dots, n]$. It is enough to compute two quantities $\mu^+(j+1)$ and $\mu^-(j+1)$, which stand for the number of input pairs in positions $j+1, j+2, \dots, n$ for which the leftmost error-contributing chain is respectively positive or negative.

It is easy and fast to compute $\mu^+(j)$ and $\mu^-(j)$ iteratively for $j = n, n-1, \dots, 1$, because for going from some $j+1$ to j we only need to consider additional leftmost carry chains, those starting with j . In the computations of the various $\mu^+(j)$'s and $\mu^-(j)$'s each carry chain needs to be considered as leftmost once only: a carry chain starting from j increments $\mu^+(j)$ if its error is positive and it increments $\mu^-(j)$ if its error is negative. In consequence, the full algorithm operates in the number of steps stated.

We have not discussed how to determine $\text{EC}(C)$'s. Knowing them is of particular interest when a designer needs to decide how to allocate the delays to the various components of the pseudo-adder to optimize the desired performance metrics while staying within some resources budget, such as energy. Although computing $\text{EC}(C)$'s is not a part of mathematical modeling, we briefly discuss it. What one presumably needs to do is to examine how the carries of C actually propagate in a specific pseudo-adder under consideration. The actual procedure depends on the design, the testing, and the computational tools available.

An error of the carry chain is by Definition 3 just an error for any one out of a well-defined set of input pairs. So a straightforward way to determine $\text{EC}(C)$ is to rely on that definition, picking a convenient pair of inputs generating C , and examining the behavior of the pseudo-adder for their addition. As treating a single C is likely to involve effort growing with n , in our asymptotic analysis characterized by the number of steps, the work required for one step is not constant, but still likely $O(n)$.

4.5 Formulation using probabilities

It is possible to rewrite (8) using probabilities, as was done in [1]. By overloading our notation, we will write $\text{EC}(i, j)$ for the error contributed by the

i - j carry chain. Let

$$p_{ij}^\sigma = \frac{1}{2^{2n}} \nu^\sigma(C) \quad \text{for } \sigma \in \{+1, -1\}.$$

Then

$$\text{Er}_{\text{avg}} = \sum_{1 \leq i \leq j \leq n} (p_{ij}^+ - p_{ij}^-) \text{EC}(i, j).$$

Formulation using probabilities allows convenient handling of input distribution other than the uniform one. We do not pursue this further here.

5 Extension to other statistical measures

5.1 Mean squared error

As before, let $\nu(C)$ denote the number of input pairs (a, b) generating C and let $\nu(C_1, C_2)$ for $C_1 \neq C_2$ denote the number of input pairs (a, b) generating both C_1 and C_2 .

Theorem 3. *The mean squared error, the expectation of $(\text{Er}(a, b))^2$, is*

$$\sum_C \nu(C) \cdot (\text{EC}(C))^2 + \sum_{C_1 \neq C_2} \nu(C_1, C_2) \cdot \text{EC}(C_1) \cdot \text{EC}(C_2)$$

divided by 2^{2n} .

Proof. The sum of squared errors is

$$\sum_{a,b} \left(\sum_{C \in C(a,b)} \text{EC}(C) \right)^2,$$

which can be written as

$$\sum_{a,b} \sum_{C \in C(a,b)} (\text{EC}(C))^2 + \sum_{a,b} \sum_{C_1, C_2 \in C(a,b) \wedge C_1 \neq C_2} \text{EC}(C_1) \cdot \text{EC}(C_2),$$

and as

$$\sum_C \nu(C) \cdot (\text{EC}(C))^2 + \sum_{C_1 \neq C_2} \nu(C_1, C_2) \cdot \text{EC}(C_1) \cdot \text{EC}(C_2),$$

from which the theorem follows. \square

5.2 Maximum absolute error

This relies on an algorithmic transformation. We reduce the computation of the maximum absolute value of the error to a simple and well-studied graph problem. Construct a weighted DAG $G = (V, E)$ to specify how carry chains interact, as follows.

1. The set of vertices V is the set of all pairs (i, j) for $1 \leq i \leq j \leq n$. Vertex (i, j) corresponds to i - j carry chain.
2. The set of edges E is the set of all pairs $((i_1, j_1), (i_2, j_2))$ for $j_1 < i_2$. Such an edge indicates that the i_1 - j_1 carry chain and the i_2 - j_2 carry chain do not overlap and therefore they can both be generated by an input pair.
3. The weight $w((i, j))$ of vertex (i, j) is the error contributed by the i - j carry chain. The weight can be negative, positive, or zero.

The weight of a path in G is the sum of the weights of the vertices in it. First a simple example.

Example 3. Consider some 16-bit adder. The set of the possible chains is $\{i\text{-}j \text{ carry chain} \mid 1 \leq i \leq j \leq 16\}$, with the corresponding set of vertices $\{(i, j) \mid 1 \leq i \leq j \leq 16\}$. Since no pair of inputs can generate both the 4-8 carry chain and the 7-10 carry chain, we do not have edge $((4, 8), (7, 10))$. Since it is possible to have both the 4-8 carry chain and the 9-10 carry chain we do have edge $((4, 8), (9, 10))$.

As some pair of inputs can generate the 4-8 carry chain, the 9-10 carry chain, and the 12-14 carry chain, there is a path $(4, 8) \rightarrow (9, 10) \rightarrow (12, 14)$. And if a pair of inputs generates exactly those three chains, the error generated by this pair will be the weight of the path. This completes the example.

By formalizing Example 3 we formulate

Lemma 5. *There is a one-to-one correspondence between the paths in G and the sets of non-overlapping carry chains.*

Proof. A set σ of carry chains can appear in a sum if and only if no two carry chains in it overlap. We show that there is a bijection between the set of all such σ 's and the set of all the paths in G .

Let π be any path, say v_1, v_2, \dots, v_m , written more explicitly as $(i_1, j_1), (i_2, j_2), \dots, (i_m, j_m)$. As there must be an edge between any two consecutive vertices in the path it follows from the definition of E that $i_1 \leq j_1 < i_2 \leq j_2 < i_3 \leq j_3 < \dots < i_m \leq j_m$. Therefore, for any $1 \leq k < \ell \leq m$, $j_k < i_\ell$ holds, and therefore the carry chains corresponding to two distinct vertices in π do not overlap.

Let σ be any set of non-overlapping carry chains. Sort it to get a sequence $(i_1, j_1), (i_2, j_2), \dots, (i_m, j_m)$, such that $i_1 \leq i_2 \leq \dots \leq i_m$. Because no two consecutive carry chains in the sequence overlap, all the i_k 's are distinct and we can assume that $i_1 < i_2 < \dots < i_m$. Again, for the same reason, for every k , $1 \leq k < m$, the condition $j_k < i_{k+1}$ holds. So $((i_k, j_k), (i_{k+1}, j_{k+1})) \in E$. Therefore the sequence of the carry chains is in fact a path in G . \square

Let w_{\min} be the minimum weight of the paths and w_{\max} the maximum weight of the paths. Then

Theorem 4. *The maximum absolute value of an error is $\max\{-w_{\min}, w_{\max}\}$.*

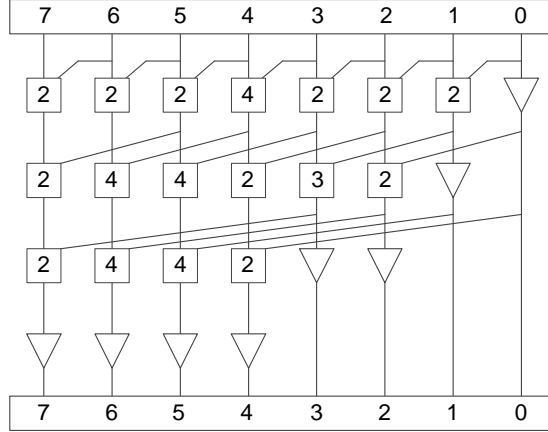


Figure 3: An 8-bit Kogge-Stone adder. The delays of the modules are written inside the corresponding nodes.

Proof. By Lemma 5, w_{\min} and w_{\max} are respectively equal to the minimum and the maximum error. Note that we needed to consider $-w_{\min}$ as there may be negative errors with large absolute values. \square

Both w_{\min} and w_{\max} can be computed fast by adapting standard all-pairs shortest path algorithms [6]. Note also that even though G has $\Theta(n^2)$ vertices, any path in it is of length $O(n)$ only, which makes computations faster than for general DAGs.

6 Example: A Kogge-Stone-based pseudo-adder

We will now look at a small pseudo-adder based on the KSA and show for it how the signs of the carry chains interact in computing the absolute value of the error produced by sets of carry chains for a pair of inputs, elaborating on Example 1. The KSA is parallel-prefix, carry-lookahead adder. Ignoring for now the numbers in the vertices, an example of an 8-bit KSA is depicted Fig. 3, which we adapt from Weste and Harris [7], where also a more detailed description of such figures is given.

For us what is important is that the KSA operates with some parallelism. Therefore, given some allocation of propagation delays, it may happen that both $c_{k-1} = 1$ and $c_k = 1$ but c_k propagated to its final destination faster than c_{k-1} did to its final destination. We note here that irrespective of the allocation of propagation delays in the RCA, that condition can never happen in its simple architecture, and that is why its mathematical modeling could be so much simpler.

The numbers in the vertices representing modules in Fig. 3 are the delays introduced in those modules. In Fig. 4, we flesh out Example 1 for this

		Position									
		7	6	5	4	3	2	1	0		
<i>a</i>	→	0	0	1	0	1	0	1	1	0	
<i>b</i>	→	0	0	0	1	1	1	0	1	1	

Time									s'	$s - s'$
0	0	0	0	0	0	0	0	0	0	+145
1	0	0	1	1	0	1	1	0	109	+36
2	0	0	1	1	0	1	1	0	109	+36
3	0	0	1	1	0	1	1	0	109	+36
4	0	0	1	1	0	1	0	0	105	+40
5	0	0	1	1	0	0	0	0	97	+48
6	0	0	1	1	0	0	0	0	97	+48
7	0	1	1	1	0	0	0	0	225	-80
8	0	1	1	1	1	0	0	0	241	-96
9	0	1	1	1	1	0	0	0	241	-96
10	0	1	0	0	1	0	0	0	145	0

Figure 4: An example of addition in an overclocked Kogge-Stone adder. Carry chains are highlighted at the instances of time in which they contribute errors.

adder. We label the two carry chains, C_1 for the 2-4 carry chain and C_2 for the 5-7 carry chain. The correct sum is obtained at time $t = 10$. We next consider the case where the output is read at the time $T = 7$.

Based on the topology of the KSA and on the delays in Fig. 3, we compute the values of s' at various instances of time. Note that sometimes the total error is positive and sometimes negative. At time T , both of the carry chains contribute errors, and we compute the total absolute error produced at that time.

1. C_1 is an error-contributing carry chain but not the leftmost one, and it contributes an error of +16.
2. C_2 is the leftmost error-contributing carry chain. That is, it is $C_L(a, b)$, and it contributes an error of -96, so

$$\widehat{\text{sgn}}(a, b) = \text{sgn}\left(\text{EC}(C_L(a, b))\right) = -1.$$

Therefore, at time $T = 7$, $|s - s'| = (-1)(-96) + (-1)(+16) = +80$.

Despite its correct hardware, the adder was incorrect in this case because it was overclocked. A simple modification of this example produced by changing the topology of the adder, would produce a pseudo-adder in which the outputs for some inputs never become correct.

7 Conclusions

The method we have presented can be applied to a wide class of circuits implementing approximate adders. As shown in [1] for the restricted case of ripple-carry adders, it is now possible to evaluate alternative schemes of allocation of voltages to the adder's components to optimize the quality of its outputs under some statistical measure. Returning to Fig. 3, the delays in the modules are a function of the voltages applied, which in turn determine the energy consumption. A key component of an optimization procedure is the ability to generate a good estimate of errors, and our method provides the mathematical means enabling that.

Acknowledgment

Some of the results presented here appeared first in a preliminary form in [8].

References

- [1] Z. M. Kedem, V. J. Mooney, K. K. Muntimadugu, and K. V. Palem, "An approach to energy-error tradeoffs in approximate ripple carry adders," in *Proceedings of the International Symposium on Low Power Electronics and Design*, August 2011, pp. 211–216.
- [2] A. Lingamneni, C. Enz, J.-L. Nagel, K. Palem, and C. Piguet, "Energy parsimonious circuit design through probabilistic pruning," in *Proceedings of the Design, Automation & Test in Europe Conference*, March 2011, pp. 764–769.
- [3] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, 2nd ed. Oxford University Press, 2010.
- [4] C. Babbage, *Passages From the Life of a Philosopher*. Longman, Green, Longman, Roberts, & Green, 1864.
- [5] A. W. Burks, H. H. Goldstine, and J. von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument," The Institute for Advanced Study, Princeton, N.J., USA, June 1946.
- [6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. MIT Press, 2009.
- [7] N. H. E. Weste and D. M. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Addison-Wesley, 2011.
- [8] Z. M. Kedem and K. K. Muntimadugu, "A general method for energy-error tradeoffs in approximate adders," Department of Computer Science, New York University, Tech. Rep. TR-2013-952, March 2013.